Universidad Nacional de Cuyo

Facultad de Ingeniería

Lic. en Ciencias de la Computación

Inteligencia Artificial II

Modelos de Aprendizaje Automático para Detección de Fake News

Cairo, Lucía - 13030

Noviembre 2024

1. INTRODUCCIÓN..... 3
2. DESCRIPCIÓN..... 3
2.1. Objetivos..... 3
2.2. Dataset..... 3
3. MARCO TEÓRICO..... 3
3.1. Aprendizaje Supervisado..... 3
3.2. Técnicas de Representación Textual..... 4
3.3. Modelos de Clasificación..... 4
4. IMPLEMENTACIÓN..... 5
4.1. Importar las bibliotecas necesarias..... 5
4.2. Cargar el dataset..... 5
4.3. Explorar y limpiar los datos..... 6
4.4. Preparar los datos..... 8
4.5. Dividir los datos..... 9
4.6. Modelos..... 9
4.6.1 Regresión Logística..... 10
4.6.2 Naive Bayes..... 11
4.6.3 K-Nearest Neighbors..... 11
4.6.4 Transformers..... 12
5. COMPARACIONES..... 16
6. CONCLUSIONES..... 19
7. REFERENCIAS..... 19

1. INTRODUCCIÓN

En la actualidad, la propagación de noticias falsas representa un desafío significativo en el ámbito de la comunicación digital, afectando la confianza pública y la toma de decisiones informadas. Este trabajo se centra en el desarrollo de modelos de aprendizaje automático supervisado para detectar noticias falsas, utilizando un conjunto de datos provenientes de Kaggle. A través de la implementación de diferentes algoritmos, como Regresión Logística, Naive Bayes, K-Nearest Neighbors y Transformers, se busca comparar su desempeño y analizar su capacidad para abordar este problema. El objetivo principal es identificar estrategias efectivas para mejorar la precisión en la clasificación de noticias falsas, contribuyendo a la mitigación de la desinformación en entornos digitales.

2. DESCRIPCIÓN

El objetivo principal es desarrollar un modelo que pueda distinguir entre noticias falsas y noticias reales utilizando técnicas de clasificación supervisada. Esto ayudará a mitigar la propagación de información errónea y permitirá tomar decisiones más informadas.

2.1. Objetivos

- Implementar diferentes modelos de clasificación para identificar noticias falsas.
- Evaluar el desempeño de los modelos utilizando métricas estándar.
- Interpretar y analizar los resultados para obtener conclusiones.

2.2. Dataset

El dataset empleado proviene de Kaggle, titulado Fake News Detection. Contiene información detallada de noticias clasificadas como "reales" o "falsas", y se compone de las siguientes columnas principales:

- id: Identificador único de cada noticia.
- title: Título de la noticia.
- author: Autor o fuente de la noticia.
- text: Contenido principal de la noticia.
- label: Etiqueta que indica si la noticia es real (1) o falsa (0).

El dataset contiene 44.898 registros y fue elegido por su relevancia en el campo de la clasificación de texto y su disponibilidad pública para investigaciones y experimentos.

3. MARCO TEÓRICO

3.1. Aprendizaje Supervisado

El aprendizaje supervisado es una subárea del aprendizaje automático en la que el modelo se entrena con datos etiquetados. Consiste en aprender una función que mapea entradas a

salidas a partir de ejemplos de entrenamiento. En este contexto, el objetivo es clasificar noticias como verdaderas o falsas.

En el entrenamiento el modelo aprende patrones en los datos a partir de ejemplos con etiquetas conocidas. Se utiliza un conjunto de datos de entrenamiento que contiene pares de entrada-salida. El modelo ajusta sus parámetros para minimizar el error entre las predicciones y las etiquetas reales. Luego en la evaluación, una vez entrenado, el modelo se valida con un conjunto de datos de prueba, distinto al de entrenamiento, para medir su capacidad de generalización. Esto implica evaluar qué tan bien puede predecir correctamente datos que no ha visto antes.

3.2. Técnicas de Representación Textual

La representación de texto es un paso crucial en tareas de clasificación textual. Estas técnicas transforman datos textuales en vectores numéricos que los modelos pueden procesar.

Bag-of-Words (BoW): Representa un texto como una bolsa de palabras, ignorando el orden de las mismas. Cada documento se transforma en un vector basado en la frecuencia de las palabras. Entre sus ventajas tenemos sencillez y eficacia en tareas básicas. Como desventaja tenemos que este método pierde el contexto y las relaciones semánticas entre palabras.

Transformers: Utilizan mecanismos de atención para capturar relaciones contextuales profundas entre palabras en un texto. Modelos como BERT (Bidirectional Encoder Representations from Transformers) y GPT (Generative Pre-trained Transformer) han revolucionado el procesamiento de lenguaje natural. Los transformers manejan secuencias largas y relaciones complejas. Tienen alta demanda computacional y de recursos.

3.3. Modelos de Clasificación

Existen variedad de modelos de clasificación, a continuación se detallan los utilizados para la clasificación de noticias.

Regresión Logística: Es un modelo lineal que estima la probabilidad de una clase utilizando la función sigmoide.

- Ventajas: Simple, interpretable y rápido.
- Limitaciones: Ineficiente para relaciones no lineales.

Naive Bayes: Es un clasificador probabilístico basado en el teorema de Bayes. Supone independencia entre las características.

- Ventajas: Rápido y eficaz para datos textuales.
- Limitaciones: La suposición de independencia puede no ser realista.

K-Nearest Neighbors (KNN): Clasifica datos basándose en la similitud con sus vecinos más cercanos.

- Ventajas: Simple y no paramétrico.
- Limitaciones: Ineficiente con datos de alta dimensionalidad.

Redes Neuronales: Inspiradas en la estructura del cerebro humano, las redes neuronales son modelos no lineales que pueden aprender relaciones complejas en los datos. Arquitecturas comunes: Redes densas, convolucionales y recurrentes.

- Ventajas: Adaptabilidad a grandes cantidades de datos y tareas complejas.
- Limitaciones: Requieren un gran volumen de datos para entrenar eficazmente. Alto costo computacional y demanda de recursos.

4. IMPLEMENTACIÓN

Se realizo un notebook de jupyter en el lenguaje de programacion python.

4.1. Importar las bibliotecas necesarias

Primeramente se introducen las herramientas necesarias para desarrollar el proyecto. Aqui se incluyen bibliotecas para manipulación de datos, visualización, procesamiento de texto, modelado y evaluación de resultados.

4.2. Cargar el dataset

El proyecto trabaja con dos conjuntos de datos que representan noticias falsas y verdaderas. En esta etapa, se detalla el proceso de carga, exploración y unificación de los mismos.

Los conjuntos de datos se cargan desde archivos CSV utilizando pandas. Esto genera dos estructuras DataFrame: el conjunto ‘df_fake’ que tiene 23.481 registros, y ‘df_true’ con 21.417 registros. Ambos poseen 4 columnas (title, text, subject, date), todas de tipo object y ningun conjunto tiene valores nulos, lo que asegura que el análisis inicial no requiere imputación de datos.

Luego se selecciona solo la columna text, ya que es la necesaria para el análisis, eliminando aquellas que no aportan directamente al objetivo. También se agrega una columna label para identificar si una noticia es falsa (fake) o verdadera (true), lo que permite combinar los datos más adelante de forma diferenciada.

Finalmente ambos conjuntos se combinan y mezclan en un único DataFrame, lo que reduce la posibilidad de errores durante el análisis y para aplicar los mismos pasos de limpieza y transformación a todas las noticias, independientemente de su categoría. El dataset combinado contiene 44.898 registros.

	title	text	label
0	Ben Stein Calls Out 9th Circuit Court: Committ...	21st Century Wire says Ben Stein, reputable pr...	fake
1	Trump drops Steve Bannon from National Securit...	WASHINGTON (Reuters) - U.S. President Donald T...	true
2	Puerto Rico expects U.S. to lift Jones Act shi...	(Reuters) - Puerto Rico Governor Ricardo Rosse...	true
3	OOPS: Trump Just Accidentally Confirmed He Le...	On Monday, Donald Trump once again embarrassed...	fake
4	Donald Trump heads for Scotland to reopen a go...	GLASGOW, Scotland (Reuters) - Most U.S. presid...	true

Figura 1: Muestra del conjunto de datos combinados con el que se trabajara

4.3. Explorar y limpiar los datos

En esta sección se llevó a cabo una exploración inicial de los datos, seguida de procesos de limpieza para garantizar la calidad del conjunto de datos final.

Eliminación de duplicados

Se implementó una función para eliminar filas duplicadas basadas en el contenido de la columna de texto. Esto permitió reducir el conjunto de datos de 44.898 registros a 38.646, eliminando un total de 6.252 registros duplicados.

Análisis de la distribución de etiquetas

Se analizó la distribución de las clases "fake" y "true" en el conjunto de datos. La visualización mostró que ambas están balanceadas, lo que es ideal para evitar problemas de desbalance que podrían afectar el rendimiento de los modelos de clasificación.

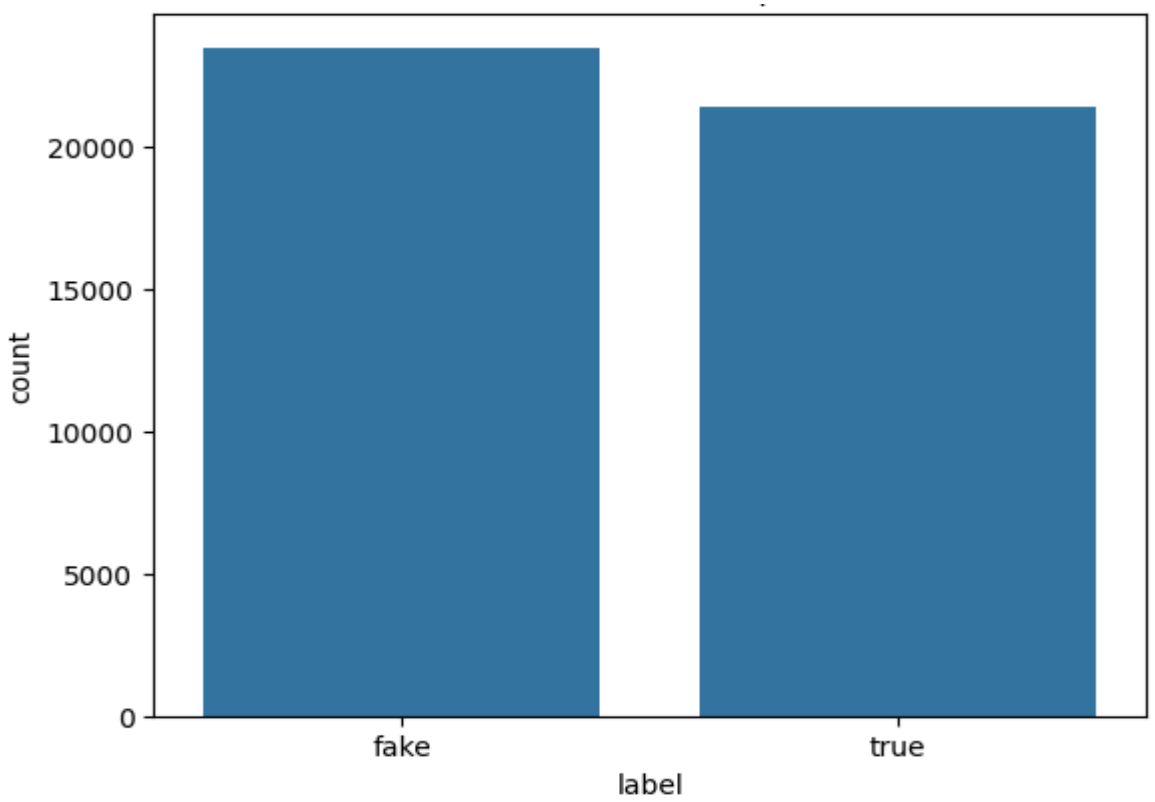

Figura 2: Distribución de las clases "fake" y "true" en el conjunto de datos

Longitud de los textos

Se calculó la longitud de los textos en términos del número de caracteres. La distribución de la longitud de los textos se visualizó por clase, mostrando que las noticias falsas y verdaderas tienen patrones similares en cuanto a longitud, con la mayoría de los textos concentrados en un rango de longitud promedio y algunos pocos casos atípicos más extensos.

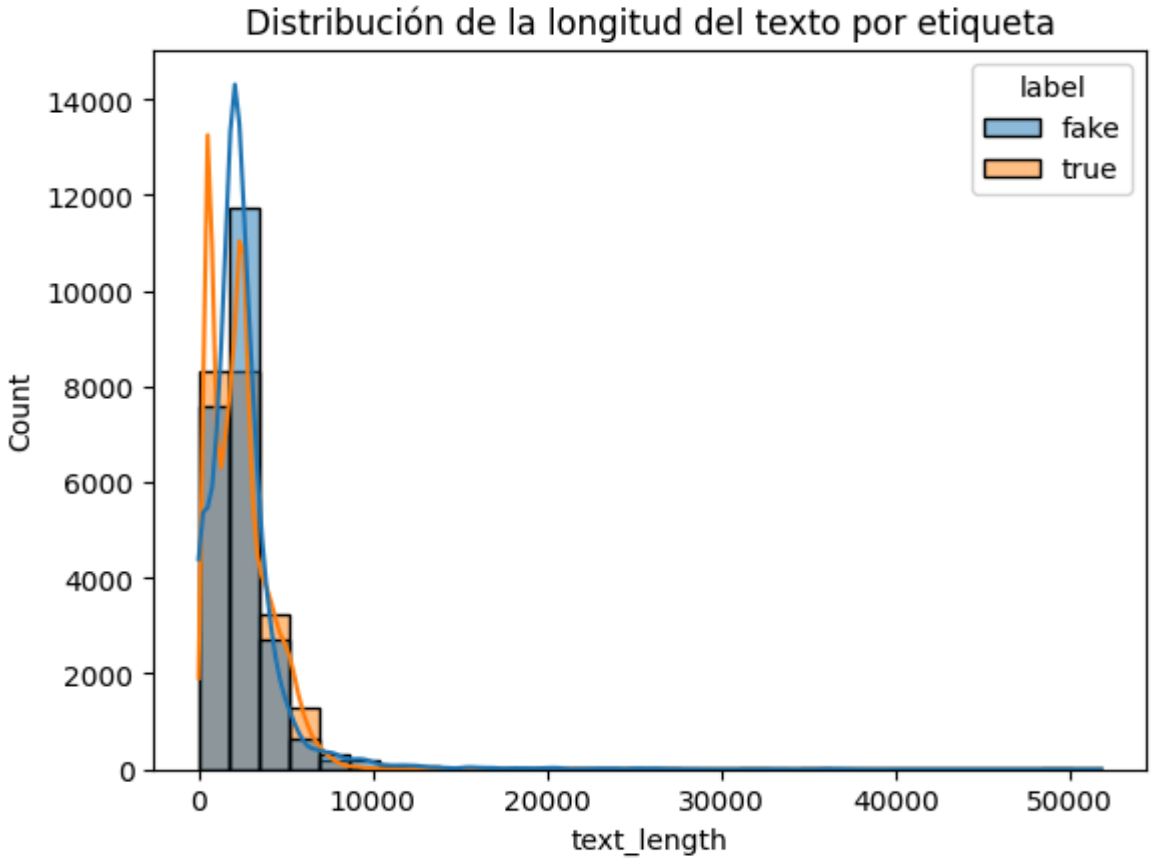

Figura 3: Distribución de la longitud de texto (caracteres) por etiqueta

También se calculó la longitud de los textos en términos del número de palabras. Se visualizó que los textos etiquetados como fake tienden a ser más largos que los etiquetados como true. Esto se refleja en los valores atípicos más frecuentes y extremos en la categoría fake. La similitud en la mediana de ambas etiquetas indica que, para la mayoría de los casos, la longitud de los textos es comparable entre fake y true.

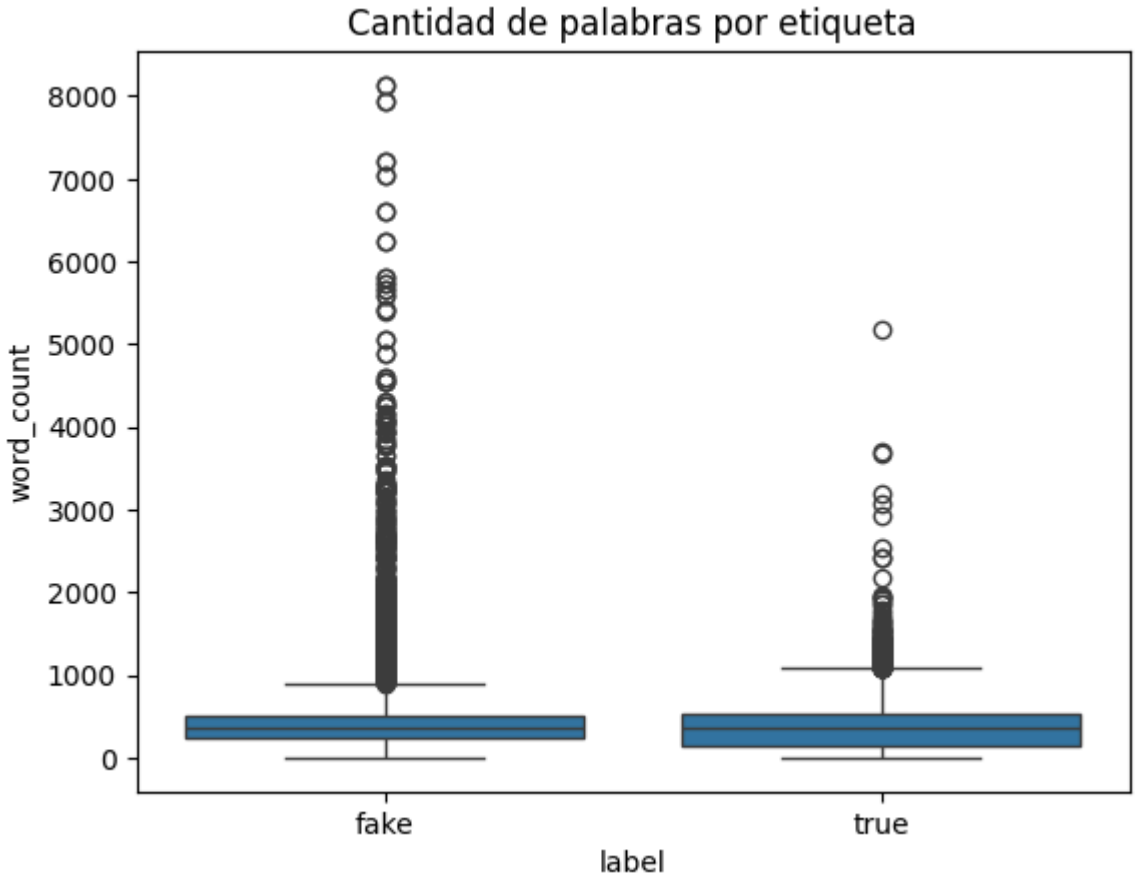

Figura 4: Distribución de la longitud de texto (palabras) por etiqueta

Análisis de palabras más comunes

Utilizando técnicas de procesamiento de texto, se identificaron las palabras más frecuentes en el corpus después de eliminar palabras vacías (stopwords). Entre las palabras más comunes se encuentran términos como "Trump", "said", y "U.S.". Esto indica la presencia de temas recurrentes y puede ser útil para comprender la semántica subyacente de los datos.

4.4. Preparar los datos

Para la preparación de los datos se realizó una normalización de etiquetas donde las etiquetas categóricas (fake y true) se convierten en valores numéricos binarios: 0 para fake y 1 para true. Luego se realiza el preprocesamiento del texto donde todo el texto se convierte a minúsculas para eliminar diferencias debidas a capitalización y se eliminan todos los caracteres que no sean letras, números o espacios.

	text	label
0	21st century wire says ben stein reputable pro...	0
1	washington reuters us president donald trump r...	1
2	reuters puerto rico governor ricardo rossello ...	1
3	on monday donald trump once again embarrassed ...	0
4	glasgow scotland reuters most us presidential ...	1

Figura 5: resultado del dataset preprocesado

Como resultado, la columna text ahora contiene texto limpio y en un formato consistente que puede ser tokenizado o vectorizado para el modelo y la columna label es binaria, facilitando tareas de clasificación supervisada.

4.5. Dividir los datos

En esta sección, se detalla el proceso de división de los datos en conjuntos de entrenamiento y prueba, un paso crucial para evaluar el desempeño del modelo en datos no vistos durante el entrenamiento.

Primero, se separan las características (texto) y las etiquetas en las variables X e y, respectivamente. Se utiliza la función train_test_split de sklearn para dividir los datos en un conjunto de entrenamiento y otro de prueba. Las proporciones seleccionadas son 80% para entrenamiento y 20% para prueba. Así, el conjunto de entrenamiento quedó con 35.918 registros, mientras que el conjunto de prueba con 8.980 registros.

Luego, se analiza la distribución de las etiquetas en los conjuntos para verificar que el balance de clases se haya mantenido correctamente.

Conjunto	Clase 0 (Fake)	Clase 1 (True)	Relación Fake/True
Original	52.30%	47.70%	~1.10
Entrenamiento	52.30%	47.70%	~1.10
Prueba	52.29%	47.71%	~1.10

Figura 6: proporciones por etiqueta de cada conjunto

Aunque las clases no tienen exactamente la misma cantidad de ejemplos, la diferencia es lo suficientemente pequeña para que las etiquetas se consideren balanceadas en términos prácticos.

4.6. Modelos

En esta sección, se detalla la implementación de los modelos creados: regresión logística, naive bayes, KNN y red neuronal con transformers.

Para los primeros tres modelos mencionados, se utiliza la representación Bag-of-Words para los datos preprocesados. Este enfoque convierte el texto en una matriz de características que puede ser utilizada por el modelo para realizar la clasificación.

Se utiliza la clase CountVectorizer de scikit-learn para transformar los datos de texto en una representación Bag-of-Words:

```
vectorizer = CountVectorizer()
X_train_bow = vectorizer.fit_transform(X_train)
X_test_bow = vectorizer.transform(X_test)
```

Esta transformación convierte cada texto en un vector de frecuencias de palabras, donde cada entrada representa el número de veces que una palabra aparece en el texto.

4.6.1 Regresión Logística

Se implementa el modelo de regresión logística y se entrena este con los datos de entrenamiento transformados con Bag-of-Words.

```
lr_model = LogisticRegression()
lr_model.fit(X_train_bow, y_train)
```

Asi, con 203677 términos (tamaño del vocabulario), el modelo aprende a asociar patrones en las características Bag-of-Words con las etiquetas correspondientes.

Finalmente se realiza la predicción de las etiquetas en el conjunto de prueba y se evalúa el rendimiento del modelo mediante métricas estándar.

```
Accuracy - Logistic Regression: 0.9973273942093541
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4696
           1       1.00      1.00      1.00      4284

    accuracy                           1.00      8980
   macro avg       1.00      1.00      1.00      8980
weighted avg       1.00      1.00      1.00      8980
```

Figura 7: Reporte de clasificación del modelo de regresión logística.

El modelo logra un desempeño excelente con una exactitud del 99.73% en el conjunto de prueba. Las métricas de precisión, recall y f1-score son perfectas para ambas clases (0 y 1), indicando que el modelo clasifica correctamente tanto noticias falsas como verdaderas.

Para este caso particular en que las métricas son casi perfectas es importante verificar si el rendimiento del modelo es realista o si hay señales de overfitting. Para esto se revisó el desempeño del modelo en el conjunto de entrenamiento y también se usó validación cruzada (con 5 validaciones cruzadas) para evaluar el modelo en diferentes divisiones de los datos. Se obtuvo que la precisión promedio en validación cruzada (0.9970) y del

entrenamiento (1.0) están en línea con la precisión en el conjunto de prueba, lo que indica que el modelo aprendió patrones útiles en los datos, en lugar de memorizar ejemplos específicos y no hay signos evidentes de sobreajuste.

4.6.2 Naive Bayes

También se implementa el modelo de Naive Bayes y se entrena con los datos de entrenamiento transformados con Bag-of-Words.

```
nb_model = MultinomialNB()
nb_model.fit(X_train_bow, y_train)
```

Luego se realiza la predicción y evalua el rendimiento del modelo.

```
Accuracy - Naive Bayes: 0.9650334075723831
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      4696
           1       0.95      0.97      0.96      4284

    accuracy                           0.97      8980
   macro avg       0.96      0.97      0.96      8980
weighted avg       0.97      0.97      0.97      8980
```

Figura 8: Reporte de clasificación del modelo de Naive Bayes.

El modelo logra una exactitud del 96.5% en el conjunto de prueba. Las métricas de precisión, recall y f1-score son ligeramente mejores para la clasificación de noticias falsas.

4.6.3 K-Nearest Neighbors

Similar a la implementación de los dos modelos presentados anteriormente, el modelo de KNN se crea y entrena con los datos de entrenamiento transformados con Bag-of-Words.

```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train_bow, y_train)
```

También se realiza la predicción de las etiquetas en el conjunto de prueba y se evalúa el rendimiento del modelo mediante métricas mencionadas.

```
Accuracy - KNN: 0.8983296213808464
              precision    recall  f1-score   support

           0       0.91      0.89      0.90      4696
           1       0.88      0.91      0.90      4284

    accuracy                           0.90      8980
   macro avg       0.90      0.90      0.90      8980
weighted avg       0.90      0.90      0.90      8980
```

Figura 9: Reporte de clasificación del modelo knn.

El modelo tiene exactitud del 89.83% en el conjunto de prueba, siendo esta la menor de todos los modelos implementados.

4.6.4 Transformers

El último modelo implementado es el más complejo, basado en la arquitectura de Transformers. Esta sección describe el desarrollo del modelo, las técnicas empleadas en el preprocesamiento de datos, la arquitectura del modelo, el proceso de entrenamiento y los resultados obtenidos.

Preprocesamiento de los datos

El preprocesamiento de datos es una etapa crucial para asegurar que los datos estén en el formato adecuado para ser procesados por el modelo.

Se utilizó el Tokenizer de Keras para convertir los textos en secuencias de enteros. Cada palabra del texto es convertida en un entero único basado en su frecuencia en el conjunto de datos. Además, dado que las secuencias de texto tienen longitudes variables, se rellenaron todas las secuencias con ceros hasta alcanzar una longitud fija de 200 palabras. Esto asegura que las entradas al modelo tengan una forma consistente.

Luego, el conjunto de datos fue dividido en tres subconjuntos para garantizar un entrenamiento, validación y prueba adecuados:

- Entrenamiento: 60% de los datos, utilizados para ajustar los pesos del modelo.
- Validación: 20% de los datos, empleados para evaluar el rendimiento del modelo durante el entrenamiento.
- Prueba: 20% de los datos, reservados para evaluar el rendimiento final.

Se verificó que todos los conjuntos estuvieran balanceados entre la cantidad de noticias falsas y verdaderas.

Arquitectura del modelo

La arquitectura del modelo está basada en el diseño clásico de los Transformers, que combina mecanismos de atención y redes neuronales feed-forward. A continuación, se describen sus componentes principales:

El bloque principal del modelo es el TransformerBlock, que incluye los siguientes elementos:

- Atención Multi-Cabezal: Este mecanismo permite que el modelo enfoque diferentes partes de la secuencia simultáneamente. Es configurado con 2 cabezales de atención (num_heads=2) con una dimensión de incrustación de 32 (embed_dim=32).

- Red Feed-Forward: Consiste en dos capas densas, la primera capa con activación ReLU y la segunda capa con dimensión de salida igual a la entrada del bloque (32).

- Normalización y Dropout: Tiene capas de normalización (LayerNormalization) que estabilizan el aprendizaje y de Dropout con una tasa del 10% (rate=0.1) para prevenir sobreajuste.

A diferencia de otros modelos basados en Transformers, como BERT y GPT, que cuentan con un preentrenamiento en grandes cantidades de datos no supervisados, este Transformer no tiene ningún tipo de entrenamiento previo. Esto implica que los pesos iniciales del modelo son aleatorios, y el aprendizaje depende completamente de los datos del conjunto de entrenamiento proporcionado. Este enfoque simplifica la implementación, pero puede requerir más datos etiquetados y un entrenamiento cuidadoso para alcanzar un rendimiento óptimo.

Se implementó una capa personalizada TokenAndPositionEmbedding que combina incrustaciones de tokens (representan cada palabra en un espacio vectorial según su significado) e incrustaciones de posición (capturan la información sobre el orden de las palabras en la secuencia). Esto permite que el modelo distinga entre palabras con el mismo significado pero en posiciones diferentes.

La arquitectura final del modelo incluye los siguientes componentes:

1. Capa de entrada para secuencias de longitud 200.
2. Capa de incrustaciones (TokenAndPositionEmbedding).
3. Bloque Transformer (TransformerBlock).
4. Capa de Pooling global (GlobalAveragePooling1D) para reducir la dimensionalidad.
5. Capas densas adicionales:
    - capa con 10 unidades y activación ReLU.
    - capa de salida con 1 unidad y activación sigmoide para clasificación binaria.

Entrenamiento del modelo

El modelo se compiló con el optimizador Adam (configuración predeterminada), la función de pérdida Binary Crossentropy, adecuada para tareas de clasificación binaria, y como métrica la precisión (accuracy) para evaluar el rendimiento. Además, se utilizó EarlyStopping para detener el entrenamiento si la pérdida de validación no mejora en 6 épocas consecutivas.

```
Epoch 1/50
842/842 ━━━━━━━━━━━━━━━━━━━━ 103s 117ms/step - accuracy: 0.8533 - loss: 0.2940 - val_accuracy: 0.9775 - val_loss: 0.0584
Epoch 2/50
842/842 ━━━━━━━━━━━━━━━━━━━━ 141s 116ms/step - accuracy: 0.9915 - loss: 0.0349 - val_accuracy: 0.9869 - val_loss: 0.0451
Epoch 3/50
842/842 ━━━━━━━━━━━━━━━━━━━━ 138s 111ms/step - accuracy: 0.9962 - loss: 0.0140 - val_accuracy: 0.9867 - val_loss: 0.0635
Epoch 4/50
842/842 ━━━━━━━━━━━━━━━━━━━━ 140s 109ms/step - accuracy: 0.9957 - loss: 0.0116 - val_accuracy: 0.9871 - val_loss: 0.0680
Epoch 5/50
842/842 ━━━━━━━━━━━━━━━━━━━━ 144s 112ms/step - accuracy: 0.9980 - loss: 0.0052 - val_accuracy: 0.9873 - val_loss: 0.0606
Epoch 6/50
842/842 ━━━━━━━━━━━━━━━━━━━━ 146s 117ms/step - accuracy: 0.9980 - loss: 0.0064 - val_accuracy: 0.9892 - val_loss: 0.0636
Epoch 7/50
842/842 ━━━━━━━━━━━━━━━━━━━━ 137s 111ms/step - accuracy: 0.9982 - loss: 0.0049 - val_accuracy: 0.9878 - val_loss: 0.0789
Epoch 8/50
842/842 ━━━━━━━━━━━━━━━━━━━━ 144s 113ms/step - accuracy: 0.9983 - loss: 0.0055 - val_accuracy: 0.9888 - val_loss: 0.0541
```

Figura 10: Historial del entrenamiento del modelo con Transformers.

El modelo fue entrenado durante 8 épocas con el objetivo de optimizar la precisión en el conjunto de entrenamiento y validación.

Como se observa en la figura 10, el modelo comenzó con una precisión de 85.33% en la primera época. Al final de la octava época, la precisión en el conjunto de entrenamiento alcanzó 99.83%. Mientras que la precisión en el conjunto de validación mostró una mejora progresiva, alcanzando un máximo de 98.92% en la sexta época.

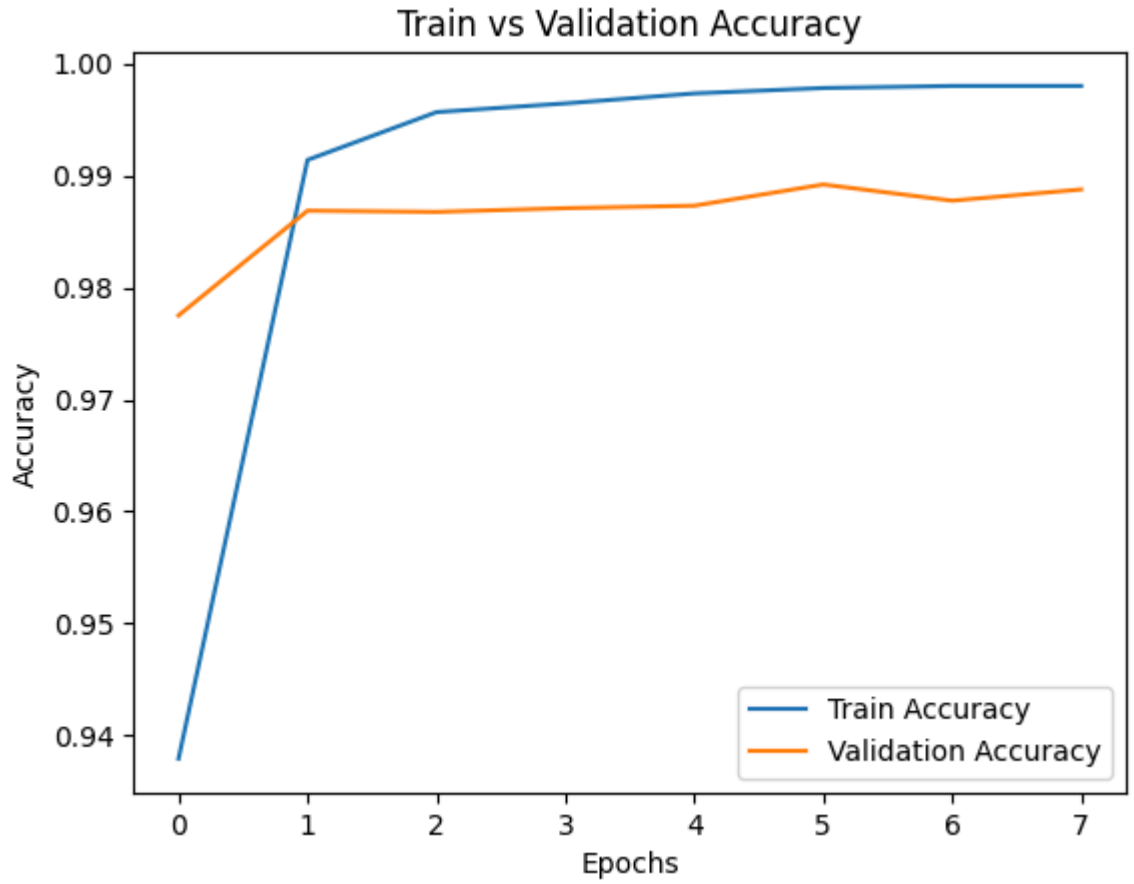

Figura 11: gráfica de precisión en train y validation a lo largo de las épocas.

La gráfica de precisión muestra que el modelo se aproxima al 100% de precicion con una leve diferencia entre entrenamiento y validación, ambos con resultados prometedores.

A partir de la segunda época, la precisión en el entrenamiento mejora drásticamente, alcanzando 99.15%, y se mantiene alrededor del 99.8% hasta la octava época. Esto sugiere que el modelo está aprendiendo rápidamente los patrones del conjunto de entrenamiento, y su capacidad de clasificación mejora de manera constante. Por otro lado, la precisión en validación también aumenta, pero más lentamente. Se estabiliza en torno al 98.8% entre la cuarta y octava época. Es importante notar que no alcanza la misma precisión que el conjunto de entrenamiento, lo que sugiere que el modelo está empezando a memorizar los datos de entrenamiento (sobreajuste leve).

En la figura 10, también podemos evaluar la pérdida en ambos conjuntos. El conjunto de entrenamiento disminuyó de 0.2940 en la primera época a 0.0055 en la octava época. En validación, la pérdida inicial fue de 0.0584, con fluctuaciones en las últimas épocas, alcanzando 0.0541 en la octava época.

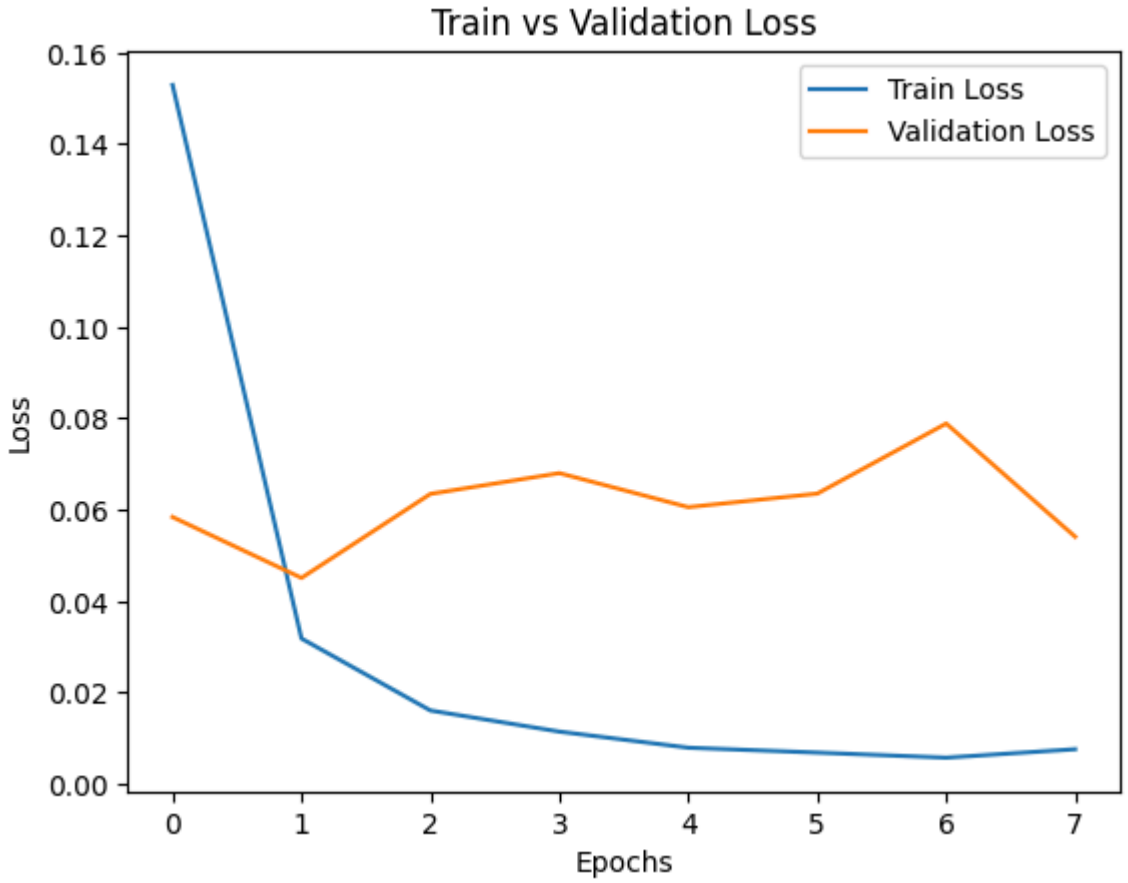

Figura 12: gráfica de pérdida en train y validation a lo largo de las épocas.

La gráfica indica que durante el entrenamiento, la pérdida inicial fue mayor en el conjunto de entrenamiento (0.2940) que en validación (0.0584), indicando un inicio menos eficiente. Sin embargo, la pérdida en entrenamiento cayó drásticamente desde la segunda época, estabilizándose por debajo de 0.01 al final, reflejando un aprendizaje progresivo. En contraste, la pérdida en validación mostró fluctuaciones y una disminución menos consistente, alcanzando 0.0541 en la octava época, lo que sugiere una generalización limitada. La marcada diferencia entre ambas pérdidas confirma un ligero sobreajuste, donde el modelo memorizó características específicas del entrenamiento en lugar de capturar patrones generales.

En resumen, el modelo logra una alta precisión en entrenamiento y validación, lo cual demuestra su capacidad para ajustarse al conjunto de datos. Existe un gap (brecha) pequeño entre la pérdida de entrenamiento y la pérdida de validación a partir de la tercera época. A partir de la octava época, los resultados no muestran mejoras significativas en validación, lo que sugiere que un entrenamiento adicional no aportaría mejoras

sustanciales. El modelo alcanza un rendimiento sobresaliente con una precisión cercana al 99% en validación. Sin embargo, se observa sobreajuste leve después de la cuarta época.

Evaluación del modelo

Finalmente, el modelo fue evaluado en el conjunto de prueba, obteniendo:

- Precisión: 98.88%.
- Pérdida: 0.045.

Por lo que el modelo basado en Transformers demuestra una capacidad notable para resolver el problema.

5. COMPARACIONES

En esta sección se presentan los resultados comparativos entre los diferentes modelos evaluados. Los resultados incluyen métricas de precisión (accuracy) y matrices de confusión tanto en su versión absoluta como normalizada, las cuales permiten un análisis más detallado del rendimiento de cada modelo.

Modelo	**Accuracy**
Logistic Regression	99,73%
Naive Bayes	96,5%
KNN	89,08%
Transformers	98,88%

Figura 13: resultados comparativos entre modelos.

Logistic Regression

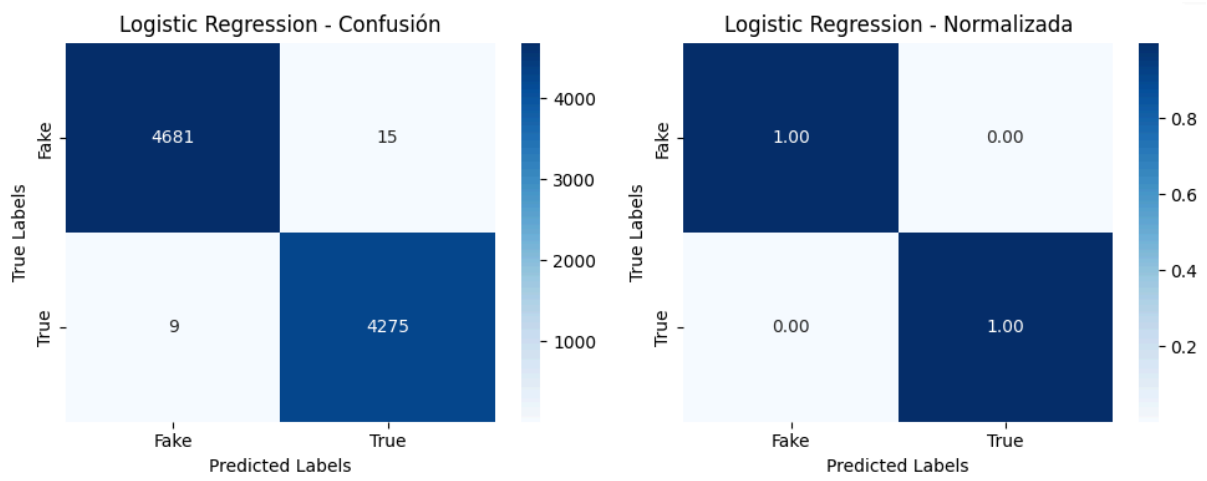

Figura 14: matrices de confusión del modelo de regresión logística.

La matriz muestra que la Regresión Logística predice casi perfectamente las clases:

- 4681 verdaderos negativos (Fake correctamente clasificados como Fake).
- 4275 verdaderos positivos (True correctamente clasificados como True).
- Solo 15 falsos positivos y 9 falsos negativos, valores extremadamente bajos.

Los valores normalizados confirman una precisión del 100% para ambas clases, con tasas de falsos positivos y falsos negativos de prácticamente 0%.

Así, la Regresión Logística presenta un rendimiento sobresaliente y casi perfecto. La alta precisión y la ausencia de errores relevantes indican que el modelo es altamente eficiente para distinguir entre clases Fake y True.

Naive Bayes

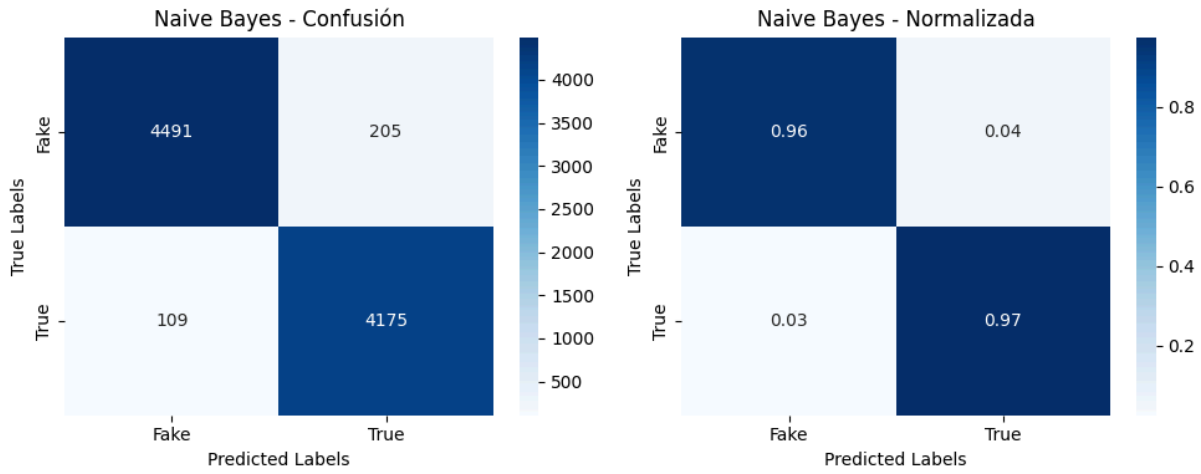

Figura 15: matrices de confusión del modelo de naive bayes.

Se obtuvo:

- 4491 verdaderos negativos
- 4175 verdaderos positivos
- Sin embargo, se observan 205 falsos positivos (Fake clasificados incorrectamente como True) y 109 falsos negativos (True clasificados como Fake).

La clase Fake tiene una tasa de acierto del 96%, con un 4% de falsos positivos y la clase True una precisión del 97%, con solo 3% de falsos negativos.

Aunque el rendimiento de Naive Bayes es bueno, presenta más errores que la Regresión Logística, especialmente en falsos positivos. Este comportamiento puede deberse a la fuerte suposición de independencia entre las características que hace el algoritmo, lo que limita su capacidad de capturar relaciones complejas entre los datos.

K-Nearest Neighbors (KNN)

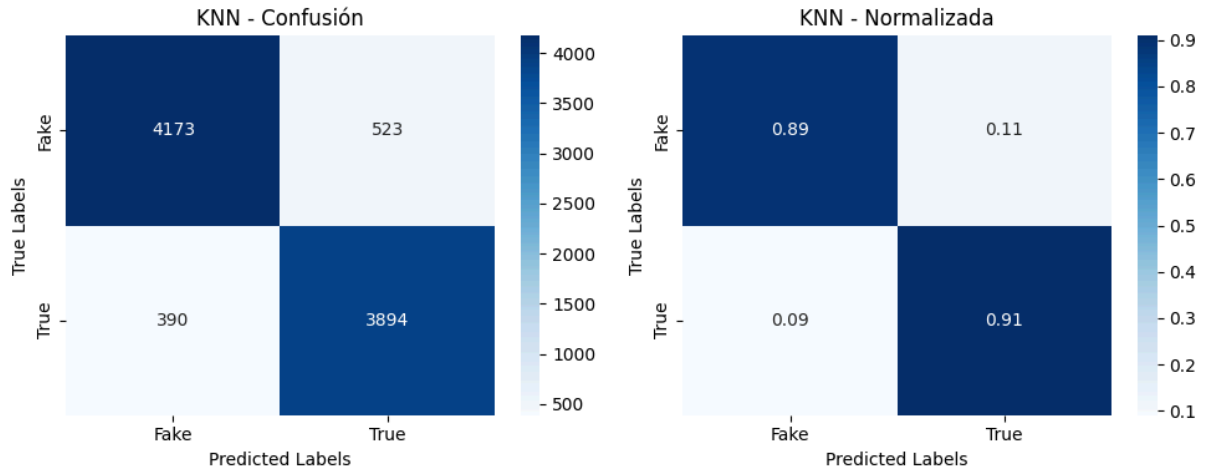

Figura 16: matrices de confusión del modelo de knn.

La figura muestra que para knn obtenemos:

- 4173 verdaderos negativos
- 3894 verdaderos positivos.
- Sin embargo, se observan 523 falsos positivos (11%) y 390 falsos negativos (9%).

La clase Fake tiene una precisión del 89%, mientras que la clase True alcanza el 91%. Los errores son significativamente mayores en comparación con los modelos anteriores.

El modelo KNN tiene el peor rendimiento de los cuatro modelos evaluados, con una precisión considerablemente menor y un número elevado de errores. Este comportamiento puede explicarse por la naturaleza del algoritmo KNN, que es muy sensible a la distribución de los datos y a la presencia de ruido. La cantidad de falsos positivos y falsos negativos sugiere que KNN no generaliza bien en este problema.

Transformers

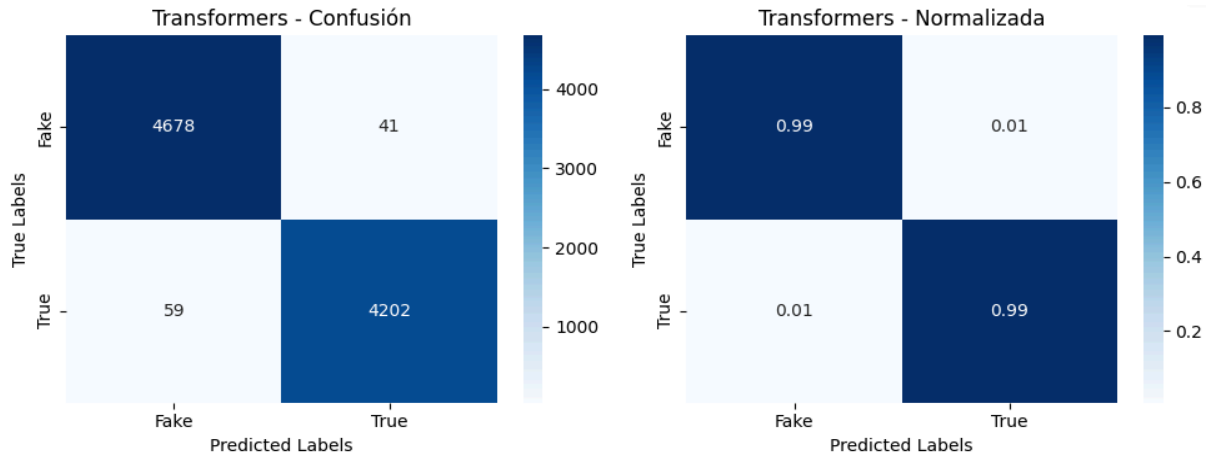

Figura 17: matrices de confusión del modelo basado en transformers

Este modelo muestra:

- 4678 verdaderos negativos
- 4202 verdaderos positivos.
- Solo 41 falsos positivos (1%) y 59 falsos negativos (1%).

Ambas clases tienen una precisión del 99%, con errores mínimos del 1%.

El modelo basado en Transformers muestra un desempeño excelente, con resultados muy cercanos a los de la Regresión Logística. La capacidad de los Transformers para capturar relaciones complejas entre los datos permite obtener una precisión alta con una mínima tasa de error.

Conclusión General

- La Regresión Logística es el modelo con mejor desempeño, logrando una precisión casi perfecta (99.73%) y errores insignificantes en ambas clases.
- El modelo Transformers también alcanza resultados sobresalientes, lo que refleja su capacidad para manejar datos complejos y lograr predicciones precisas.
- Naive Bayes presenta un buen rendimiento general, aunque con errores más evidentes en comparación con Logistic Regression y Transformers.
- El modelo KNN tiene el rendimiento más bajo debido a su sensibilidad a la distribución de los datos y al ruido.

6. CONCLUSIONES

El análisis comparativo destaca que la Regresión Logística es el modelo más efectivo, logrando una precisión del 99.73% gracias a su capacidad para clasificar con alta exactitud las noticias como verdaderas o falsas. Los Transformers, con una precisión del 98.85%, también muestran un rendimiento sobresaliente, siendo más robustos y adecuados para problemas complejos, aunque con un mayor costo computacional. En contraste, Naive Bayes (96.50%) y KNN (89.08%) presentan un desempeño considerablemente menor, con limitaciones en la captura de relaciones complejas y un número significativo de errores en el caso de KNN. Estos resultados subrayan la importancia de seleccionar modelos avanzados y bien ajustados según la naturaleza del problema y los recursos disponibles.

7. REFERENCIAS

- Sampat, The Real. *Fake News Detection*. Kaggle, 2024. Disponible en: https://www.kaggle.com/code/therealsampat/fake-news-detection/notebook .
- Cátedra de Inteligencia Artificial II. Diapositivas de clase. Facultad de Ingeniería, Universidad Nacional de Cuyo, 2024.
- Keras Team. Text Classification with Transformer. Keras Examples, 2024. Disponible en: https://keras.io/examples/nlp/text_classification_with_transformer/ .

- Chollet, François. Deep Learning with Python, 2nd Edition. Manning Publications, 2021.
- OpenAI. ChatGPT, Modelo de lenguaje basado en inteligencia artificial, utilizado como herramienta de apoyo en el análisis y desarrollo del presente informe, 2024.